INF1006 Computer Networks

Assignment 2

Submission Deadline: 28 March 2024, 11.59p.m.

Socket Programming: Developing a Chat Application Using Python

Aim

The project aims to expose you to the world of internet application development using socket programming in Python. In particular, we will make use of socket library to develop a simple chat application. This is a group assignment. Students are expected to work with the same group members they worked with during Assignment 1.

Methodology

Instead of developing the application from nothing, we are going to use existing codes found on the Internet. Although there are many web sites that are dedicated to teaching socket programming in Python, for this assignment, you will be given the server and client Python codes, which are downloadable from the xsite LMS. ***Your group is required to make extensions to the codes based on the requirements highlighted below***.

The aim of this project is to familiarize yourself with socket programming using Python as the base programming language. The server-side chat application is programmed to handle multiple instances of clients connecting to it. Correspondingly, the client application is designed to connect to the server.

Essentially, the application is comprised of two parts: server and client. The server is designed to support multiple clients. All messages from a client are sent to the server for processing and distribution to other clients. In a typical setup, there will be only one server, while multiple clients can be connected to the server.

In this project, you are required to make several extensions to the codes. The extensions include:

1. When an instance of client is executed, in addition to requesting the IP address of the server and its port number, it is required to prompt the user for a username, e.g. John. When it successfully connects to the server, the server will return a message that reads "[Welcome John!]" to be displayed on the console. For the rest of the connected clients, the client's console will receive and display the message "[John joined]". Notice that all messages generated by the server are enclosed in square brackets […].
2. If another instance of a client attempts to use the same name, say John, the server returns a message that reads "[Username has already been used. Please enter another name.]". In which case, the client will prompt the user to re-enter another name.
3. Each time a client sends a message, the server will append the sender's username to the message and broadcast it to all other users. For example, if John sends a message "Should we wrap up this project?", the server will broadcast the message "[John:] Should we wrap up this project?".
4. When the user enters "@quit", the client will be disconnected from the server. At the same time, messages will be sent to all connected clients informing them that the user has quit. For

example, if John quits the chat (by issuing "@quit"), the message "[John exited]" will be sent to all the rest of the clients.

5. When the user enters "@names", the server will send back all the users' names currently connected to the server. For example, if John, Peter and Stella are connected, the server will return "[Connected users: John, Peter, Stella]" (note, ordering of names is not important).
6. A user can send a personal message to a specific user by appending the message with @xxx, where xxx is the name of the user. For example, if Stella wants to send a personal message to John, she can type "@John Are you free next Tuesday?". In this case, only John will receive the message.
7. Insert comments on the codes whenever necessary to make it easier to understand.

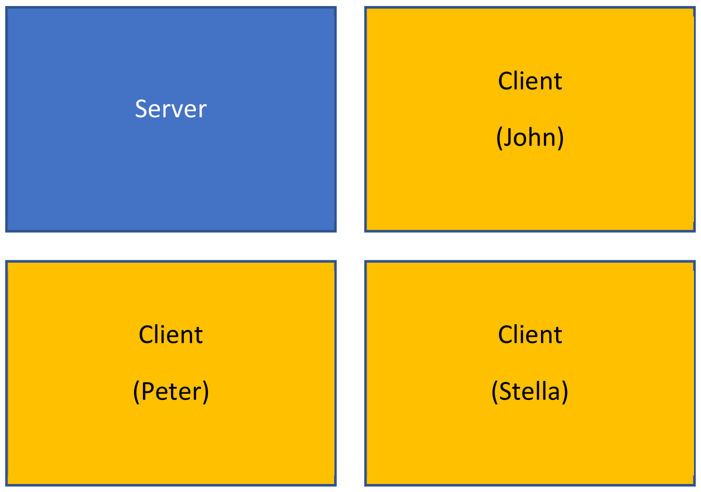
Optional Extension

1. A user can create a group by using the command "@group set ggg xxx, yyy, zzz" where ggg is the group name, while xxx, yyy and zzz are the names in the group. For example, "@group set dinner John, Stella" will create a group named dinner with John and Stella as the users in the group. The group name must contain only alphanumeric characters and in one word.
2. If a group is successfully created, a message will be sent from the server to all the members in the group. In the above example, if dinner is successfully created, the server will send the message "[You are enrolled in the dinner group]" to John and Stella. Only John and Stella can send messages to the group using @group send xxx, where xxx is the message
3. Any member in a group can delete the group by issuing the command "@group delete ggg" where ggg is the group name.
4. Any member in a group can choose to leave the group by issuing the command "@group leave ggg" where ggg is the group name.

Deliverables

The following are the deliverables for this assignment:

1. Zip the project folder containing the server and client codes (under the folder `codes`). Name the server as server.py, while the client as client.py. Name the zip file under your group name, example G1.zip, if your group is group 1. Include a Readme.txt file in the zip file that lists the members of your group, including name, student ID and email.
2. A narrated screen capture video (preferably in mp4 and name it as video.mp4) that demonstrates the working of the codes. In the video, arrange four consoles to occupy the entire screen. The top left console is the server, while the other three consoles are the clients as shown. Place the video under the folder `video`.
3. Only the leader in the group need to submit to the LMS.



4. In the video, show how each client connects to the server starting with John, followed by Peter.

5. For the third client (bottom right), attempt to connect the server using the username “John”. If it fails (since username “John” has been used), attempt to connect using the username “Stella”.

6. Show how each client sends messages to the rest.

7. Using John’s client, issue the command “@names”. The server should return all the usernames connected to the server.

8. Using John’s client, send a personal message to Stella with the following content “@Stella Are you available next Tuesday?” (or any other messages). Correspondingly, Stella will reply to John in a similar fashion.

9. Finally, issue the command “@quit” starting with John, followed by Peter and Stella.

10. Limit the video to at most 3 minutes.

11. Name the video under your group number and include it together with the zip file in (1) and upload it to LMS.

[OPTIONAL] If your group decides to include the extensions, use the same setup as above to demonstrate step-by-step the various functions implemented, including setting a group, sending message within a group, leaving a group and deleting a group. Also, your program needs to take care of exceptions such as invalid group name, duplicate group name, invalid user name, etc. It is your group’s responsibility to demonstrate (via the video) as comprehensively as possible the features built into the system.

Late Submission

A penalty of 20% per day of this assignment marks (including Sunday and public holiday) will be imposed for late submission unless extension has been granted by the tutor prior to the submission date. Request for extension will be granted on a case-by-case basis. Any work submitted more than 4 days after the submission date will not be accepted and no mark will be awarded.

Notes on Plagiarism

The University's policy on copying does not allow you to copy software as well as your assessment solutions from another person. Copying of another person work is unacceptable. It is the responsibility of all students that their assessment solutions are their own work. You must also ensure that others do not obtain access to your solutions for copying a part of them. Where such plagiarism is detected, both assessments involved will receive ZERO mark.

The Usage of AI Tools

- You may use AI programs e.g., ChatGPT to help generate ideas and brainstorm. However, you should note that the material generated by these programs may be inaccurate, incomplete, or otherwise problematic. Beware that using ite may also potentially stifle your own independent thinking and creativity.
- You may not submit any work generated by an AI program as your own. If you include material generated by an AI program, it should be cited like any other reference material (with due consideration for the quality of the reference, which may be poor).
- Any plagiarism or other form of cheating will be dealt with severely under relevant SIT policies.
- The SIT policy on using generative AI tools can be found here: https://xsite.singaporetech.edu.sg/d2l/le/enhancedSequenceViewer/91270?url=https%3A%2F%2Feb7d8702-81cc-4acd-8156-b72a5f076aca.sequences.api.brightspace.com%2F91270%2Factivity%2F633460%3FfilterOnDatesAndDepth%3D1